Обработка и Агрегирование Табличных Данных

Пилипенко А.О., Банников Д.А. гр.311, ВМК МГУ

Постановка задачи

1. Формулирование примера, используемого для описания задачи.
2. Написание программы, выполняющей обработку табличных данных.
    a. Математическая интерпретация задачи и его решения.
    b. Описание основных функций, используемых при программном решении.
3. Описание кода программы, полученного в результате решения построенного примера.

Описание задачи

ApplePen - это большая торговая сеть, которая занимается продажей всего двух продуктов: яблок и карандашей. Ее магазины расположены в различных уголках Соединенных Штатов и более 10 лет обслуживают покупателей. Недавно топ-менеджмент компании решил более активно использовать имеющиеся у них данные в принятии решений. Каждый магазин собирает информацию о:

1. Закупках (поставки яблок и карандашей два раза в месяц),
2. Продажах (лог транзакций, по записи на каждую проданную позицию),
3. Инвентарь (месячные данные общего количества яблок и карандашей на складе).

Данные сохранены в формате CSV. Внутри файла данные отсортированы по дате. К сожалению, данные никогда не консолидировались и не проверялись. Нам необходимо получить следующие данные в CSV-файлах:

1. Состояние склада на каждый день

Данные о состоянии склада в конце каждого дня после того как все поставки и продажи были совершены. Подобная информация будет очень ценна менеджерам магазинов. Состояние склада должно строится на основе месячных данных об инвентаре. Известно, что люди воруют из магазинов, но сейчас нет никакой возможности узнать объем ворованного товара.

2. Месячные данные о количестве ворованного товара

3. Агрегированные данные об объемах продаж и количестве ворованной продукции по штату и году.

Математическое решение задачи

Рассмотрим для начала предприятие единого товара. Упрощение актуально, так как все транзакции фиксируются независимо, все данные о товарах хранятся раздельно, уместно считать, что обработка данных об одном товаре будет проводиться независимо от происходящих транзакций другого товара, а все

вычисления будут схожими. Для того, чтобы математическое решение выглядело нагляднее, чем три формулы, взявшиеся из нашей головы, используем приближенное к реальности описание решения.

Представим, что мы работает в бухгалтерии описанной выше компании. Начался новый рабочий месяц. Приходят данные со склада: на складе находится СКЛАД единиц товара, пусть это будут яблоки. Представим, что за первый день произошло 3 сделки, в результате которых были куплены столько единиц товара: ПОКУПКА 1, ПОКУПКА 2 и ПОКУПКА 3. На данный момент нас не интересует, в каких торговых точках были совершены покупки. Но мы можем точно узнать (без учета возможности краж), сколько товара должно остаться на складе к концу дня:

СКЛАД_ЗАВТРА = СКЛАД - (ПОКУПКА 1 + ПОКУПКА 2 + ПОКУПКА 3)

Аналогичные вычисления будут на следующий день, и на следующий после него. И вот настал первый день поставок: на склад пришли ПОСТАВКИ. Тогда к концу дня, с учетом всевозможных покупок за день (ПОКУПКИ) состояние склада изменится так:

СКЛАД_ЗАВТРА = СКЛАД + ПОСТАВКИ - ПОКУПКИ

Первая задача не так уж и сложна, а что со второй задачей: сборе данных о ворованном товаре? Дождемся последнего дня месяца. Целый месяц мы подсчитывали, сколько товара должно остаться на складе, и вот мы несем отчет начальству: “К концу месяца на складе осталось СКЛАД_РЕЗУЛЬТАТ товара”. Но начальство в ярости. Вы перепроверяете все отчеты и рассчеты, но никак не можете найти ошибку - ее нет. И когда вы в слезах спрашиваете начальство, в чем же ваша ошибка, они протягивают вам бумагу с надписью: “Отчет со склада № ***. На складе осталось СКЛАД_РЕАЛЬНО товара”. Фирма-поставщик товара очень тщательно проверяет объем поставок, так что они ошибиться не могли. Так и есть: со склада УКРАЛИ = СКЛАД_РЕЗУЛЬТАТ - СКЛАД_РЕАЛЬНО товара. Ничего не поделаешь, придется вести отдельную отчетность по кражам. Так как отчеты со склада приходят каждый месяц, отчетность по кражам вести можно вести с той же частотой или реже. К тому же, не стоит забывать, что теперь надо изменить и наши данные в базе подсчета ежедневного состояния склада:

СКЛАД_РЕЗУЛЬТАТ = СКЛАД_РЕАЛЬНО

Разобравшись с кражами, мы уверенно начинаем новый рабочий месяц. Но приходят указания от начальства: “Требуется агрегировать данные об объемах продаж и количестве ворованной продукции по штату и году”. У вас уже отваливается челюсть, вы бежите за словарем, чтобы узнать, чего же такого хочет начальство?

Агрегация, или агрегирование (лат. aggregatio «присоединение») — процесс объединения элементов в одну систему.

Теперь они хотят получить годовые отчетности по полученным нами данным в отдельности для каждого штата, где расположены торговые точки. Ничего не поделаешь, придется описанные выше расчеты провести отдельно для каждого штата, и составлять новые отчетности. Единственное радует, готовить их надо только раз в году.

Основы работы с CSV-файлами

Полученные теоретические решения осталось применить к имеющимся у нас данным, а именно: CSV-файлам. Для начала работы с CSV файлами в среде *python* нам потребуются библиотеки: *numpy* и *pandas*.

Библиотека *numpy* очень популярна, благодаря возможности выполнять различные математические операции с нестандартными для системы данными - массивами, списками и т.п. Функционал данной библиотеки интуитивно понятен, потому каждую функцию из данной библиотеки, используемую в нашем коде, мы не будет описывать сейчас.

Библиотека *pandas* используется для анализа данных, в частности, она позволяет нам работать с файлами CSV. Для этого используются функции:

read_csv(adr) - чтение данных из CSV файла, расположенного по адресу adr.

X.set_index(‘date’, inplace=True) - использование столбца date для индексирования данных.

X.join(Y, lsuffix='_A', rsuffix='_B') - объединение двух наборов данных X и Y по индексам (создаются новые индексы, а к именам старых приписываются суффиксы).

.reset_index() - восстановление стандартной индексации.

X.date.map(lambda date : date[:4]) - замещение содержимого столбца дат (ГГГГ-ММ-ДД) данными только года (ГГГГ).

X.groupby(‘date’) - объединение всех строк с одинаковой датой в одну.

.agg([f1, f2])- объединение всех строк, при которых выполняется условие f1 в первую, а все строки с выполнением функции f2 - во вторую строку.

X.reindex_like(Y) - замена элементов из X элементами из Y с тем же индексом.

.fillna(0) - замена NaN элементы на 0.

.shift(1) - сдвиг строк по индексам на 1 вперед.

X.sum() - суммирует элементы серии или столбца из таблицы данных.

X.append(Y) - присоединение данных Y к набору данных X.

X.to_csv(adr) - запись данных X в CSV файл, расположенный по адресу *adr*.

Код программы-решения данной задачи

Далее мы предоставляем текст программы, написанной нами для решения описанной выше задачи. Среда разработки - Python, используемые библиотеки описаны выше. Надо добавить, что входные данные для программы были распределены так, что для каждого штата определена тройка файлов: ST-inventory.csv, ST-supply.csv, ST-sell.csv (ST - State). Программа работает с такими уникальными тройками, автоматически определяя их в указанной на вход папке.

Данные в файлах имеют такой вид:

Inventory(Склад):

Date, Apple, Pen. (Дата указана в формате ГГГГ-ММ-ДД, последнее число месяца. Количество находящегося товара на складах указано в единицах, товар: яблоки и ручки соответственно)

Supply(Поставки):

Date, Apple, Pen. (аналогично дата указана в формате ГГГГ-ММ-ДД, поставки происходят два раза в месяц - 1 и 15 числа. Данные о товаре указаны в том же формате, что использовался на складе)

Sell(Продажи):

Date, Sku_Number. (Дата формата ГГГГ-ММ-ДД - день совершения продажи. SKU_Number - (Stock Keeping Unit) Код складской единицы, содержит информацию о штате, где совершалась продажа, название товара и уникальный код идентификации сделки)

Программа ориентирована на запуск через консоль: “python program.py INPUT_DIR OUTPUT_DIR”. В результате работы программы в указанную папку будут созданы файлы: ST-daily.csv, ST-steal.csv, states.csv, содержащие данные о состоянии склада на каждый день, количестве ворованного товара к концу месяца - для каждого штата, и суммарный годовой отчет по всем штатам соответственно.

Код программы:

***

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#Считает, что изначально склад пуст, так как для первого дня не предоставлены данные о состоянии склада

import pandas as pd
import numpy as np
import sys
from os import listdir, makedirs
from os.path import isfile, join, exists

def apple(daily):
    return np.sum(daily == 'a')
def pen(daily):
    return np.sum(daily == 'p')

def export_stat(stats):
    return pd.concat(stats).groupby(['year','state']).sum()

def stores_story(sell, supply, inventory, sell_trans):
    sell.sku_num = sell.sku_num.map(sell_trans)
    sell_daily = sell.groupby('date')['sku_num'].agg([apple, pen])
    changes_daily = supply.reindex_like(sell_daily).fillna(0) - sell_daily
# расширяем фрейм на каждый день
    monthly_sum_change = changes_daily.groupby(lambda date: date[:-2]).agg(np.cumsum)
# заполняя поле состоянием склада в последний день предыдущего месяца
    ex_inventory = inventory.reindex_like(monthly_sum_change).shift(1).fillna(axis=0, method='ffill').fillna(0)
    store_daily_st = ex_inventory + monthly_sum_change
    stolen_month = store_daily_st.reindex_like(inventory) - inventory
```

```
    return store_daily_st, stolen_month, sell_daily.groupby(lambda date: date[:-2]).sum()

if __name__ == '__main__':
    #входные файлы перечисляются аргументами в командной строке (аргументов два:
1 input, 2 output файлы)
    inp_dir = "in"
    out_dir = "out"
    statistics = []
    sell_trans = lambda transaction: transaction[6]

    all_files = [f for f in listdir(inp_dir) if isfile(join(inp_dir, f))]
    stores = [f[:-8] for f in all_files if f[-8:] == "sell.csv"]
    if not exists(out_dir):# если не существует выходного файла, создаём
        makedirs(out_dir)

    for store in stores:
        print ("Processing...")
        inventory = pd.read_csv(inp_dir + '/' + store + 'inventory.csv')
        sold = pd.read_csv(inp_dir + '/' + store + 'sell.csv')
        sold_const = pd.read_csv(inp_dir + '/' + store + 'sell.csv')
        supply = pd.read_csv(inp_dir + '/' + store + 'supply.csv')

        supply.set_index('date', inplace=True)
        inventory.set_index('date', inplace=True)

        state, stolen, sell = stores_story(sold, supply, inventory, sell_trans)

        state.to_csv(out_dir + '/' + store + 'daily.csv')
        stolen.to_csv(out_dir + '/' + store + 'steal.csv')

        sell.index = stolen.index
        stats = sell.join(stolen, lsuffix='_sold', rsuffix='_stolen').reset_index()

        stats['state'] = sold_const.sku_num.map(lambda sku_num: sku_num[6:8])

        stats['year'] = stats.date.map(lambda date: date[:4])
        del stats['date']
        statistics.append(stats)

    export_stat(statistics).to_csv(out_dir + '/states.csv')
    print("Finish!")
***
```